Exploring the Space of IR Functions

Parantapa Goswami, Simon Moura, Eric Gaussier, Massih-Reza Amini, Francis Maes

Exploring the Space of IR Functions

Parantapa Goswami[1], Simon Moura[1], Eric Gaussier[1], Massih-Reza Amini[1], and Francis Maes[2]

[1] Université Grenoble Alps, CNRS - LIG/AMA
Grenoble, France
firstname.lastname@imag.fr
[2] D-Labs, Paris, France
francis@d-labs.fr

Abstract. In this paper we propose an approach to discover functions for IR ranking from a space of simple closed-form mathematical functions. In general, all IR ranking models are based on two basic variables, namely, term frequency and document frequency. Here a grammar for generating all possible functions is defined which consists of the two above said variables and basic mathematical operations - addition, subtraction, multiplication, division, logarithm, exponential and square root. The large set of functions generated by this grammar is filtered by checking mathematical feasibility and satisfiability to heuristic constraints on IR scoring functions proposed by the community. Obtained candidate functions are tested on various standard IR collections and several simple but highly efficient scoring functions are identified. We show that these newly discovered functions are outperforming other state-of-the-art IR scoring models through extensive experimentation on several IR collections. We also compare the performance of functions satisfying IR constraints to those which do not, and show that the former set of functions clearly outperforms the latter one[3].

Keywords: IR Theory, Function Generation, Automatic Discovery

1 Introduction

Developing new term-document scoring functions that outperform already existing traditional scoring schemes is one of the most interesting and popular research area in theoretical information retrieval (IR). Many state-of-the-art IR scoring schemes have been developed since the dawn of IR research, such as the vector space model [19], the language model [17], BM25 [18], and, more recently, the HMM [15], DFR [1] and information-based models [2]. All these scoring schemes were developed along what one could call a "theoretical line", in which theoretical principles guide the development of the scoring function, and then the developed function is assessed on different standard IR test collections. There

[3] The program code and the list of generated scoring functions can be found on http://ama.liglab.fr/resourcestools/code/ir-functions-generation/

is however a chance, or more precisely a risk, that some high performing scoring schemes will not come into light through such an approach, as they are not so intuitive and/or are not so easily explainable theoretically. Quoting [9]: *There is no guarantee that existing ranking functions are the best/optimal ones available. It seems likely that more powerful functions are yet to be discovered.*

These considerations have led researchers to explore the space of IR functions in a more systematic way, even though such attempts have always been limited by the complexity of the search space (of infinite dimension and containing potentially all real functions) with regard to the current computational power. The first attempts to this exploration were based on genetic programming and genetic algorithms, which were seen as a way to automatically learn IR functions by exploring parts of the solution space stochastically [12, 16, 6]. [9] applied genetic programming to discover optimal personalized ranking functions for each individual query or a consensus ranking function for a group of queries. The approach shows the power of automated function discovery using machine intelligence tools. But, being a non-deterministic method, the solutions generated by these genetic programming approaches are often difficult to analyze. Moreover, there is another issue associated with genetic programming, namely "code bloat" – the fact that almost all genetic programming algorithms have a tendency to produce larger and larger functions along the iterations. Thus there is a high risk of missing simple functions of high quality. To this end, [5] provides metrics to measure the distance between rank lists generated by different solutions, and thus explaining their position is the solution space. More recently, researchers have focused on particular function forms as linear combinations or well-defined kernel functions, the parameters of which are learned from some training data. This approach has been highly successful in IR, through the various "learning to rank" methods proposed so far: pointwise approaches, e.g. [4], pairwise approaches, e.g. [3, 11, 13], or list wise approaches, e.g. [20].

Even though all the methods mentioned above enlarge the space of scoring functions, they are still limited in two aspects: first, they usually assume that the IR scoring function takes a particular form (e.g. linear or polynomial), and they require some training set in order to learn the parameters of the function given a particular collection. Two questions, directly addressed in the current study, thus remain open: (*a*) is it possible to explore the space of IR scoring functions in a more systematic (i.e. exhaustive) way? (*b*) Is it possible to find a function that behaves well on all (or most) collections, and thus dispenses from re-training the function each time a new collection is considered? To answer those questions, we introduce an automatic discovery approach based on the systematic exploration of a search space of simple closed-form mathematical functions. This approach is inspired from the work of [14] on multi-armed bandit problems and is here coupled with the use of heuristic IR constraints [10] to prune the search space and so, limiting the computational requirements. Such a possibility was mentioned in [6] but has not been tried to the best of our knowledge.

The remainder of the paper is organized as follows: Section 2 introduces the function generation process we have followed, whereas Sections 3 and 4 present the experiments and results obtained. Finally, Section 5 concludes the paper.

2 Function Generation

In this section we present the proposed exploration strategy for function generation and their validation that we deploy to find a set of candidate scoring functions. These functions assign positive scores to a document $\mathbf{d}$ and a query term $\mathbf{w}$ and are involved in the retrieval status value of a query-document pair $(\mathbf{d}, \mathbf{q})$. Two variables are at the basis of classical IR scoring functions: term frequency ($t^{\mathbf{d}}_{\mathbf{w}}$) and document frequency ($\mathcal{N}_{\mathbf{w}}$), to score a document $\mathbf{d}$ with respect to a query term $\mathbf{w}$. However it is well known that normalized versions of these variables yield better results. For example language models use relative term counts [17] and the BM25 model uses the Okapi normalization [18]. Any such normalization scheme can be used with our approach. For this work, we selected a common scheme, which is the one used in DFR and in information based models [2]. Thus, we consider the following variables (for notations see Table 1):

- Normalized term frequency $x^{\mathbf{d}}_{\mathbf{w}} = t^{\mathbf{d}}_{\mathbf{w}} \log\left(1 + c\frac{l_{avg}}{l_{\mathbf{d}}}\right)$. Here $c \in \mathbb{R}$ is a multiplying factor. This variable incorporates both $t^{\mathbf{d}}_{\mathbf{w}}$ and $l_{\mathbf{d}}$. For simplicity, unless otherwise stated it is written as x from now on;
- Normalized document frequency $y_{\mathbf{w}} = \frac{\mathcal{N}_{\mathbf{w}}}{\mathcal{N}}$. For simplicity, unless otherwise stated it is written as y from now on;
- A constant real valued parameter $k \in \mathbb{R}$.

Notation	Description
$t^{\mathbf{d}}_{\mathbf{w}}$	# of occurrences of term $\mathbf{w}$ in document $\mathbf{d}$, term frequency
$t^{\mathbf{q}}_{\mathbf{w}}$	# of occurrences of term $\mathbf{w}$ in query $\mathbf{q}$
$x^{\mathbf{d}}_{\mathbf{w}}$	normalized version of term frequency
$\mathcal{N}_{\mathbf{w}}$	# of documents in the collection containing $\mathbf{w}$, document frequency
$y_{\mathbf{w}}$	normalized version of document frequency
$\mathcal{N}$	# of documents in a given collection
$l_{\mathbf{d}}$	Length of document $\mathbf{d}$ in # of terms
l_{avg}	Average length of documents in a given collection

Table 1. Notations

We hence define a function as the combination of the basic quantities x, y and k, and unary (logarithm, exponentiation with respect to e, square root, unary negation) and binary (addition, subtraction, multiplication, division and exponentiation with respect to any real number) operations. The grammar we use to generate syntactically correct functions is given in Figure 1. The $-(.)$ signifies the unary negation operation (e.g. $-y$). Thus, a function g may be a binary expression $\mathbb{B}(g, g)$, or a unary expression $\mathbb{U}(g)$, or a symbol $\mathbb{S}$.

$$
\begin{aligned}
g &::= \mathbb{B}(g, g) \mid \mathbb{U}(g) \mid \mathbb{S} \\
\mathbb{B} &::= + \mid - \mid \times \mid \div \mid pow \\
\mathbb{U} &::= log \mid exp \mid sqrt \mid -(.) \\
\mathbb{S} &::= x \mid y \mid k
\end{aligned}
$$

Fig. 1. Grammar $\mathcal{G}$ to generate scoring functions

After a first combination of these quantities and operations we look at the validity of the generated functions. This validity verification is mainly three fold.

- *Domain of definition*, where we verify that all the operations used in a function are well defined. Bad operations include logarithm or square root of a negative number and division by zero.
- *Positiveness*, where we check that a generated function is positive valued. In fact, under all normal circumstances raw term and document frequency values are strictly positive. Hence $x \in \mathbb{R}$ and $x > 0$. Moreover, $\mathcal{N}_{\mathbf{w}} \leq \mathcal{N}$, which gives $y \in \mathbb{R}$ and $0 \leq y \leq 1$
- *IR constraints*, where we look if the generated function satisfies or not the heuristic IR constraints proposed by the community [10, 2]. That is for the generated function g, if we have:

$$\frac{\partial g(x,y)}{\partial t^d_{\mathbf{w}}} > 0, \frac{\partial^2 g(x,y)}{(\partial t^d_{\mathbf{w}})^2} < 0, \frac{\partial g(x,y)}{\partial \mathcal{N}_{\mathbf{w}}} < 0, \frac{\partial g(x,y)}{\partial l_{\mathbf{d}}} < 0$$

From the definitions of x and y, it can be shown $\frac{\partial x}{\partial t^d_{\mathbf{w}}} > 0$, $\frac{\partial x}{\partial l_{\mathbf{d}}} > 0$ and $\frac{\partial y}{\partial \mathcal{N}_{\mathbf{w}}} > 0$. Hence it is sufficient that g satisfies the following constraints:

$$\frac{\partial g(x,y)}{\partial x} > 0, \frac{\partial^2 g(x,y)}{\partial x^2} < 0, \frac{\partial g(x,y)}{\partial y} < 0$$

We now define the *length* of a function as the number of symbols or operators present in that function. As for example the function $sqrt(x/y)$ has a length 4, where $sqrt()$ and the *division* are two operators and x, y are two symbols present. Similarly $sqrt(x) * exp(-y)$ has a length 6.

A function generated by grammar $\mathcal{G}$ is said to be a candidate function if it survives all the validity verification steps described earlier. Algorithm 1 specifies the iterative length-limited strategy used here to generate the set of all candidate scoring functions till length $length_{max}$ (denoted by $\mathcal{C}_V$) and it works as follows. Suppose $\mathcal{S}_{\mathcal{G}}$ is the space of all possible functions generated by grammar $\mathcal{G}$ and in a particular iteration $\mathcal{S} \subset \mathcal{S}_{\mathcal{G}}$ is the set of already generated functions with a length less than or equal to $length_{curr}$ where $length_{curr} < length_{max}$ and $\mathcal{S} = \{g_1, g_2, \ldots, g_{|\mathcal{S}|}\}$. Next iteration expands the set $\mathcal{S}$ by creating new functions. A new function $g_{|\mathcal{S}|+1}$ is created by appending another operation or symbol to any function $g_i \in \mathcal{S}$. As for example, starting from an initial empty set $\mathcal{S} = \{\}$, the function $sqrt(x/y)$ is generated by the following steps:

$$(g_1 = x) \rightarrow (g_2 = y) \rightarrow (g_3 = g_1/g_2) \rightarrow (g_4 = sqrt(g_3))$$

Once a new function $g_{|\mathcal{S}|+1}$ is generated its validity is checked. If it passes all three steps, it is included in the set of generated candidate scoring functions $\mathcal{C}_V$, otherwise it is rejected. For the purpose of our experimental study, the algorithm also stores the functions until length $length_{max}$ which do not satisfy heuristic IR constraints but otherwise are valid (denoted by $\mathcal{C}_N$).

3 Experimental Setup

We conducted a number of experiments aimed at validating those functions which respect the IR constraints and also comparing these functions with respect to classical IR models.

Algorithm 1: Generating candidate scoring functions

Input : maximum length $length_{max}$, the grammar $\mathcal{G}$
Output :
- set of candidate functions $\mathcal{C}_V$ till $length_{max}$
- set of functions which do not satisfy *heuristic IR constraints* but pass other two validity tests, $\mathcal{C}_N$

Initialization: $\mathcal{C}_V \leftarrow \{\ \}$, $\mathcal{C}_N \leftarrow \{\ \}$, $\mathcal{S} \leftarrow \{\ \}$

for $length_{curr} \in \{1, 2, \ldots, length_{max}\}$ **do**
  repeat
    A new function $g_{|\mathcal{S}|+1}$ is created by any of the following rules:
      – Append a symbol (variable or constant): $g_{|\mathcal{S}|+1} = x, y$ or k
      – Append a new unary operation: $g_{|\mathcal{S}|+1} = \mathbb{U}(g_i)$, $i \in [1, |\mathcal{S}|]$
      – Append a new binary operation: $g_{|\mathcal{S}|+1} = \mathbb{B}(g_i, g_j)$, $i, j \in [1, |\mathcal{S}|]$
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{g_{|\mathcal{S}|+1}\}$
    if $g_{|\mathcal{S}|+1}(x, y)$ *satisfies* domain of definition *test* ***AND*** $g_{|\mathcal{S}|+1}(x, y)$ *satisfies* positiveness *test* **then**
      if $g_{|\mathcal{S}|+1}(x, y)$ *satisfies* heuristic IR constraints **then**
        $\mathcal{C}_V \leftarrow \mathcal{C}_V \cup \{g_{|\mathcal{S}|+1}\}$
      else
        $\mathcal{C}_N \leftarrow \mathcal{C}_N \cup \{g_{|\mathcal{S}|+1}\}$
      end
    end
  until *all the functions till* $length_{curr}$ *in* $\mathcal{S}_\mathcal{G}$ *are generated*;
end

Algorithm 1 has been implemented using *python*(`www.python.org`). The symbolic mathematics library of python, *SymPy*(`sympy.org`) is used to symbolically verify *domain of definition*, *positiveness* and *heuristic IR constraints*. Here the maximum considered length is 8. Table 2 shows the number of candidate functions available at each length till length 8 and also the corresponding generation time. The point to be noted is that functions with length less than 4 do not pass the three steps of validity testing.

length	total number of generated functions by $\mathcal{G}$	number of candidate functions $\lvert C_V \rvert$	generation time
4	42	2	≈1 sec
5	328	10	≈1 min
6	2378	100	≈5 min
7	16447	638	≈30 min
8	49989	4657	≈1 day

Table 2. Number of candidate functions and generation times for different lengths.

The candidate functions are tested using `CLEF` (`www.clef-campaign.org`) and a large number of `TREC` (`trec.nist.gov`) collections. Basic statistics of the collections used are provided in Table 3. We appended `TREC-9` and `TREC-10` Web tracks to experiment with `WT10G`, and `TREC-2004` and `TREC-2005` Terabyte tracks for experimenting with `GOV2`.

Collection	$\mathcal{N}$	l_{avg}	Index size	#queries
GOV2	25,177,217	646	19.6 GB	100
WT10G	1,692,096	398	1.3 GB	100
TREC-3	741,856	261	427.7 MB	50
TREC-4	567,529	323	379.0 MB	50
TREC-5	524,929	339	378.0 MB	50
TREC-6,7,8	528,155	296	373.0 MB	50
CLEF-3	169,477	301	126.2 MB	60

Table 3. Statistics of various collections used in our experiments, sorted by size.

Experiments are performed on Terrier IR platform v3.5 (terrier.org) as all standard modules are integrated. We implemented our models inside this framework and used other necessary standard modules by Terrier, mainly the indexing and the evaluation components. The preprocessing steps in creating an index include stemming using Porter stemmer and removing stop-words using the stop-word list provided by Terrier. For comparison purpose three standard IR models are used, namely Okapi BM25 (denoted by BM), Dirichlet language model (denoted by LM) and log-logistic model of information model family (denoted by LG). These models are used with default parameter values, as well as optimized values. For the former, values of the parameters are the values provided as default in Terrier. That is $b = 0.75, k_1 = 1.2, k_3 = 8.0$ for BM, $\mu = 2500$ for LM and $c = 1.0$ for LG. For optimized version, the parameters are optimized from a set of values using 5 fold cross validation meaning that the query set is sequentially partitioned into 5 subsets. Of the 5 subsets, a single subset is retained for testing the model, and the remaining four subsets are used as training the parameters of each model (b and k_1 for BM, μ for LM and c for LG). The cross-validation process is then repeated 5 times, with each of the 5 subsets used exactly once for testing. Each time a query-wise average precision and precision at 10^{th} document is calculated for each set. After 5 folds, average precision of all the queries are obtained and Mean Average Precision (MAP) is calculated. Similarly average of precision at 10^{th} document for each query is obtained and average of the quantity (P@10) is reported.

4 Results

From Algorithm 1, two sets of functions are produced, namely the set of all valid candidate functions ($\mathcal{C}_V$) and the set of functions which do not satisfy heuristic IR constraints, but are valid otherwise ($\mathcal{C}_N$). We first begin our investigation over the comparison of functions within $\mathcal{C}_V$ and $\mathcal{C}_N$, and then compare functions in $\mathcal{C}_V$ with respect to the classical IR functions.

4.1 Constraint validation

From each of the sets $\mathcal{C}_V$ and $\mathcal{C}_N$, 10 subsets are created, each subset containing 100 randomly selected sample functions chosen from each initial set ($\mathcal{C}_V$ or $\mathcal{C}_N$) without replacement. These samples are tested on CLEF-3 and TREC-3,5,6,7,8. For each function MAP and P@10 are noted and they are averaged over all 100

Datasets	MAP		P@10	
	$\mathcal{C}_N$	$\mathcal{C}_V$	$\mathcal{C}_N$	$\mathcal{C}_V$
CLEF-3	0.1615	**0.3067**	0.1308	**0.2376**
TREC-3	0.0449	**0.1506**	0.0929	**0.3027**
TREC-5	0.0189	**0.0762**	0.0364	**0.1407**
TREC-6	0.1038	**0.1625**	0.1437	**0.2715**
TREC-7	0.0627	**0.1234**	0.1393	**0.2688**
TREC-8	0.0826	**0.1638**	0.1517	**0.2951**

Table 4. Average MAP and P@10 of the set of valid of $\mathcal{C}_V$ and non-valid $\mathcal{C}_N$ functions.

functions within a single sample set. Finally, average performance over 10 sample sets are reported. Table 4 shows the average `MAP` and `P@10` of 10 sample sets drawn from both $\mathcal{C}_V$ and $\mathcal{C}_N$ over these collections. As it can be seen the average `MAP` measure of functions in $\mathcal{C}_V$ are 6% to 14% higher than the `MAP` measures of functions in $\mathcal{C}_N$, and the difference is even more striking with the `P@10` measures. These results empirically validate the IR constraints, and are in line with other empirical studies which aimed to test the validity of these constraints [8, 7].

4.2 Function Validation

As shown in Table 2 there are a total of 5407 valid functions from length 4 to length 8. Testing all these functions and getting the best performing functions on all the collections is time consuming. Hence we chose a simple strategy which consists in selecting the 500 best performing functions, among 5407, on `CLEF-3` with respect to the `MAP` measure and testing these functions on the remaining datasets. We first limit our analyzes over the `TREC-3,5,6,7,8` collections. Each function is ranked based on `MAP` (and `P@10`) within a collection. An average rank of each function is estimated by taking the average of all the ranks of that function on the testing collections. Note that functions with lower average ranks are better performing. Average ranks of standard models, `BM`, `LM` and `LG` with respect to these 500 functions over the test collections are also considered. For this phase of experiment, we take the default values $k = 1.0, c = 1.0$ for the generated functions[4] and the default hyperparameter values for `BM`, `LM` and `LG`, as mentioned in the previous section. Table 5 shows the top 7 functions along with standard IR models with respect to the average rank over 5 test collections, `TREC-3,5,6,7,8`. Here we replaced k with 1 and presented the simplified functions. We denote these functions by using an exponent P or M for whether they are the best performing functions with respect to `MAP` or `P@10`, and d to indicate that they are used with their default values. We note that the 7 best ranked functions over the 5 `TREC` collections are better ranked than all the classical IR models. And that the first ranked function $(x, y) \mapsto e^{\sqrt{\log\left(\frac{x+y}{y}\right)}}$ is 2 to 6 times better ranked (with respect to `P@10` and `MAP`) than the best standard IR model. In each case we statistically compare the performance of standard models with first ranked function in terms of `MAP` using a paired two sided t-test at 0.05 level. A $\uparrow$ indicates that the corresponding model is statistically significantly worse than the first ranked function. Whereas a $\downarrow$ indicates the opposite.

[4] Note that the parameter c is used in the definition of x.

on MAP			on P@10		
functions	denoted by	rank_{avg}	functions	denoted by	rank_{avg}
$e^{\sqrt{\log\left(\frac{x+y}{y}\right)}}$	f_1^{M-d}	4.8	$e^{\sqrt{\log\left(\frac{x+y}{y}\right)}}$	f_1^{P-d}	19.8
$\sqrt{\frac{\log(1+x)}{\sqrt{y}}}$	f_2^{M-d}	14.8	$\log\left(-x+\frac{x+y}{y}\right)$	f_2^{P-d}	24.6
$\sqrt{\frac{\sqrt{xy}}{y}}$	f_3^{M-d}	15.6	$\sqrt{x+\sqrt{\frac{x}{y}}}$	f_3^{P-d}	25.2
$\sqrt{y+\sqrt{\frac{x}{y}}}$	f_4^{M-d}	17.8	$\sqrt{\sqrt{x}+\sqrt{\frac{x}{y}}}$	f_4^{P-d}	27.6
$\sqrt{\sqrt{\frac{x}{y}}.e^{-y}}$	f_5^{M-d}	18.8	$\sqrt{\frac{\log(1+x)}{\sqrt{y}}}$	f_5^{P-d}	27.8
$\sqrt{\sqrt{x}+\sqrt{\frac{x}{y}}}$	f_6^{M-d}	19.0	$\log\left(\frac{x+y}{y}\right)$	f_6^{P-d}	30.0
$\log\left(-x+\frac{x+y}{y}\right)$	f_7^{M-d}	20.0	$\log\left(\frac{x}{y}+\sqrt{e}\right)$	f_7^{P-d}	34.2
LGD	$\texttt{LG}_d$	26.0	LGD	$\texttt{LG}_d$	36.8
BM25	$\texttt{BM}_d$	108.8	BM25	$\texttt{BM}_d$	43.6
LM_{Dir}	$\texttt{LM}_d$	129.6	LM_{Dir}	$\texttt{LM}_d$	208.4

Table 5. Best functions with respect to their average ranks on `TREC-3,5,6,7,8`.

TREC-3	TREC-5	TREC-6	TREC-7	TREC-8
$\texttt{BM}_d$ (1; .252)	f_1^{M-d} (1; .140)	f_1^{M-d} (1; .249)	f_4^{M-d} (5; .194)	f_1^{M-d} (1; .256)
f_5^{M-d} (2; .252)	f_2^{M-d} (3; .139)	f_6^{M-d} (2; .248)	f_3^{M-d} (6; .194)	f_7^{M-d} (6; .255)
f_3^{M-d} (3; .250)	f_5^{M-d} (4; .138)	f_4^{M-d} (4; .247)	f_5^{M-d} (8; .194)	$\texttt{LG}_d$ (11; .255)
f_2^{M-d} (4; .249)	$\texttt{BM}_d$ (5; .138)	f_3^{M-d} (5; .247)	f_6^{M-d} (9; 194)	f_6^{M-d} (30; .252)
f_1^{M-d} (5; .249)	$\texttt{LM}_d$ (10; .137)	f_2^{M-d} (7; .246)	f_2^{M-d} (13; .193)	f_{d4}^{M} (46; .250)
f_4^{M-d} (7; .249)	f_3^{M-d} (13; .137)	$\texttt{LG}_d$ (16; .245)	f_1^{M-d} (16; .192)	f_2^{M-d} (47; .249)
$\texttt{LM}_d$ (8; .249)	f_7^{M-d} (14; .137)	f_5^{M-d} (19; .244)	f_7^{M-d} (41; .189)	f_3^{M-d} (51; .249)
f_7^{M-d} (9; .248)	f_4^{M-d} (27; .136)	f_7^{M-d} (30; .244)	$\texttt{LG}_d$ (49; .188)	f_5^{M-d} (61; .248)
f_6^{M-d} (12; .246)	$\texttt{LG}_d$ (40; .135)	$\texttt{BM}_d$ (252; .232)↑	$\texttt{LM}_d$ (64; .186)	$\texttt{BM}_d$ (181; .241)↑
$\texttt{LG}_d$ (14; .245)	f_{d6}^{M} (42; .134)	$\texttt{LM}_d$ (381; .222)↑	$\texttt{BM}_d$ (105; .183)	$\texttt{LM}_d$ (185; 0.240)↑

Table 6. `MAP` based ranks of functions with default parameter values (first value in the parenthesis) and their corresponding `MAP` (second value in the parenthesis).

In tables 6 and 7 we show the rank and respectively the `MAP` and the `P@10` measures of each function on `TREC-3,5,6,7,8` collection. Here we see that the ranks of the 7 best functions over different collections stay approximately on the same range, while the ranks of IR models may vary a lot. For example, considering the `MAP`, `BM` with its default values is ranked first on `TREC-3` while it is ranked 252^{nd} on `TREC-6` (Table 6). We find the same result by looking at the ranks of different models with respect to their `P@10` measure on different `TREC` collections (Table 7). As the language model which is ranked first on `TREC-3` and `TREC-5` ; it has the lowest rank over the three other `TREC` collections.

We now consider the top 25 functions among the best performing functions based on average ranks they have, and optimize their hyper-parameter c using 5 fold cross validation and again considering $k = 1$. In order to maintain our comparisons, we also consider the optimized versions of the standard models, `BM`, `LM` and `LG` using again 5 fold cross validation sets. Table 8 shows the top optimized

TREC-3	TREC-5	TREC-6	TREC-7	TREC-8
$\texttt{LM}_d$ (1; .532)	$\texttt{LM}_d$ (1; .276)	f_5^{P-d} (1; .418)	f_1^{P-d} (6; .432)	f_6^{P-d} (4; .474)
f_3^{P-d} (6; .516)	f_3^{P-d} (6; .248)	$\texttt{BM}_d$ (2; .414)	f_5^{P-d} (11; .430)	f_2^{P-d} (5; .474)
$\texttt{BM}_d$ (9; .514)	f_1^{P-d} (47; .236)	f_4^{P-d} (7; .402)	f_6^{P-d} (16; .428)	$\texttt{LG}_d$ (7; .474)
f_4^{P-d} (12; .506)	f_4^{P-d} (49; .234)	f_1^{P-d} (12; .400)	f_7^{P-d} (17; .428)	f_7^{P-d} (8; .472)
f_2^{P-d} (13; .504)	f_2^{P-d} (53; .234)	f_3^{P-d} (15; .398)	f_2^{P-d} (18; .428)	$\texttt{BM}_d$ (14; .472)
f_1^{P-d} (15; .504)	$\texttt{BM}_d$ (63; .232)	f_6^{P-d} (29; .396)	$\texttt{LG}_d$ (26; .428)	f_1^{P-d} (19; .468)
f_5^{P-d} (25; .496)	f_5^{P-d} (73; .228)	f_7^{P-d} (33; .396)	f_4^{P-d} (27; .426)	f_5^{P-d} (29; .460)
f_6^{P-d} (27; .496)	f_6^{P-d} (74; .228)	f_2^{P-d} (34; .396)	f_3^{P-d} (58; .422)	f_3^{P-d} (41; .458)
f_7^{P-d} (28; .496)	$\texttt{LG}_d$ (75; .228)	$\texttt{LG}_d$ (47; .396)	$\texttt{BM}_d$ (130; .418)	f_4^{P-d} (43; .458)
$\texttt{LG}_d$ (29; .496)	f_7^{P-d} (85; .226)	$\texttt{LM}_d$ (483; .346)	$\texttt{LM}_d$ (285; .392)	$\texttt{LM}_d$ (272; .432)

Table 7. P@10 based ranks of functions with default parameter values (first value in the parenthesis) and their corresponding P@10 (second value in the parenthesis).

functions along with optimized standard IR models with respect to the average rank over TREC-3,5,6,7,8. Here the exponent or the index o indicates that the function or the standard IR model are used with their optimized values.

on MAP			on P@10		
functions	denoted by	rank_{avg}	functions	denoted by	rank_{avg}
$\sqrt{\frac{\log(1+x)}{\sqrt{y}}}$	f_1^{M-o}	3.0	BM25	$\texttt{BM}_o$	9.0
$e^{\sqrt{\log\left(\frac{x+y}{y}\right)}}$	f_2^{M-o}	8.4	$\log\left(-x+\frac{x+y}{y}\right)$	f_1^{P-o}	9.2
$\sqrt{y+\sqrt{\frac{x}{y}}}$	f_3^{M-o}	8.8	$\log\left(\frac{x}{y}+\sqrt{e}\right)$	f_2^{P-o}	10.0
$\sqrt{\frac{\sqrt{xy}}{y}}$	f_4^{M-o}	9.0	$\log\left(\frac{x+y}{y}\right)$	f_3^{P-o}	10.4
$\sqrt{\sqrt{\frac{x}{y}}.e^{-y}}$	f_5^{M-o}	10.0	$\sqrt{\sqrt{x}+\sqrt{\frac{x}{y}}}$	f_4^{P-o}	11.0
$\sqrt{1+\sqrt{\frac{x}{y}}}$	f_6^{M-o}	10.4	LM_{Dir}	$\texttt{LM}_o$	12.2
$\sqrt{\sqrt{x}+\sqrt{\frac{x}{y}}}$	f_7^{M-o}	11.0	$\sqrt{x+\sqrt{\frac{x}{y}}}$	f_6^{P-o}	12.8
LM_{Dir}	$\texttt{LM}_o$	16.4	LGD	$\texttt{LG}_o$	13.0
BM25	$\texttt{BM}_o$	16.6	$\log\left(\frac{x+2y}{y}\right)$	f_6^{P-o}	13.4
LGD	$\texttt{LG}_o$	17.6	$\sqrt{1+\sqrt{\frac{x}{y}}}$	f_7^{P-o}	13.6

Table 8. Best optimized functions based on average rank on TREC-3,5,6,7,8.

In tables 9 and 10 we show detailed ranks of the optimized versions of different functions with respect to their MAP and P@10 measures on different TREC-3,5,6,7,8 collections. Although, the difference between the average ranks of the optimized IR models and the top 7 generated functions have decreased, but still, the simple generated functions are better ranked than the standard IR models. We also performed experiments on WT10G and GOV2 datasets using the same 25 selected functions found previously with the default and optimized values of their parameters. On GOV2, function 2 behaves well with a difference of 5%

in MAP with the second best model when optimizing the parameters. However on WT10G the language model seems to be the best model when using optimizing or using its default parameter values. Though it seems that the simple strategy of selecting the first 500 functions over CLEF-3 has its limits on larger collections, but the selected functions are still competitive on MAP and P@10 with respect to standard models over these large datasets.

TREC-3	TREC-5	TREC-6	TREC-7	TREC-8
$\texttt{BM}_o$ (1; .273)↓	f_1^{M-o} (1; .141)	f_1^{M-o} (1; .255)	f_5^{M-o} (4; .195)	f_1^{M-o} (1; .262)
$\texttt{LM}_o$ (2; .269)↓	$\texttt{LM}_o$ (2; .141)	f_7^{M-o} (2; .250)	f_1^{M-o} (6; .194)	f_2^{M-o} (2; .261)
f_5^{M-o} (3; .260)	f_4^{M-o} (3; .139)	f_6^{M-o} (4; .249)	f_6^{M-o} (7; .193)	$\texttt{BM}_o$ (10; .259)
f_4^{M-o} (4; .258)	f_5^{M-o} (4; .138)	f_3^{M-o} (7; .248)	f_3^{M-o} (9; .193)	$\texttt{LG}_o$ (15; .258)
f_3^{M-o} (5; .257)	f_3^{M-o} (5; .138)	f_4^{M-o} (9; .248)	f_4^{M-o} (10; .193)	f_3^{M-o} (18; .257)
f_1^{M-o} (6; .256)	f_2^{M-o} (6; .138)	f_2^{M-o} (12; .248)	f_2^{M-o} (12; .192)	f_4^{M-o} (19; .257)
f_7^{M-o} (7; .253)	f_6^{M-o} (8; .137)	f_5^{M-o} (15; .247)	f_7^{M-o} (15; .191)	f_7^{M-o} (22; .256)
f_6^{M-o} (8; .252)	f_7^{M-o} (9; .135)	$\texttt{LG}_o$ (20; .245)	$\texttt{BM}_o$ (16; .191)	f_5^{M-o} (24; .256)
f_2^{M-o} (10; .251)	$\texttt{LG}_o$ (20; .133)	$\texttt{LM}_o$ (27; .243)↑	$\texttt{LG}_o$ (20; .190)	f_{o6}^{M} (25; .255)
$\texttt{LG}_o$ (13; .246)↑	$\texttt{BM}_o$ (28; .126)↑	$\texttt{BM}_o$ (28; .232)↑	$\texttt{LM}_o$ (25; .189)	$\texttt{LM}_o$ (26; .254)

Table 9. MAP based ranks of functions with optimized parameter values (first value in the parenthesis) and their corresponding MAP (second value in the parenthesis).

TREC-3	TREC-5	TREC-6	TREC-7	TREC-8
$\texttt{BM}_o$ (1; .562)	$\texttt{LM}_o$ (1; .274)	f_4^{P-o} (1; .410)	f_1^{P-o} (1; .444)	$\texttt{BM}_o$ (1; .464)
f_5^{P-o} (2; .560)	f_5^{P-o} (2; .258)	f_1^{P-o} (2; .406)	f_3^{P-o} (3; .442)	f_2^{P-o} (3; .458)
$\texttt{LM}_o$ (3; .558)	$\texttt{BM}_o$ (4; .250)	f_3^{P-o} (6; .404)	$\texttt{LG}_o$ (4; .442)	f_7^{P-o} (4; .458)
f_4^{P-o} (7; .538)	f_4^{P-o} (9; .240)	f_2^{P-o} (8; .404)	f_2^{P-o} (5; .440)	f_3^{P-o} (5; .456)
f_1^{P-o} (17; .490)	f_2^{P-o} (13; .238)	f_6^{P-o} (9; .404)	f_7^{P-o} (6; .440)	f_6^{P-o} (7; .456)
f_3^{P-o} (19; .490)	f_7^{P-o} (14; .238)	$\texttt{LG}_o$ (11; .404)	f_6^{P-o} (7; .438)	$\texttt{LG}_o$ (9; .456)
$\texttt{LG}_o$ (20; .484))	f_1^{P-o} (15; .238)	f_5^{P-o} (15; .400)	$\texttt{BM}_o$ (13; .430)	f_1^{P-o} (11; .452)
f_2^{P-o} (21; .482)	f_3^{P-o} (19; .236)	f_7^{P-o} (18; .400)	$\texttt{LM}_o$ (17; .424)	$\texttt{LM}_o$ (13; .452)
f_6^{P-o} (22; .476)	$\texttt{LG}_o$ (21; .236)	$\texttt{BM}_o$ (26; .394)	f_5^{P-o} (18; .416)	f_4^{P-o} (19; .444)
f_7^{P-o} (25; .472)	f_6^{P-o} (22; .234)	$\texttt{LM}_o$ (27; .390)	f_4^{P-o} (19; .412)	f_5^{P-o} (27; .438)

Table 10. P@10 based ranks of functions with optimized parameter values (first value in the parenthesis) and their corresponding P@10 (second value in the parenthesis).

WT10G		GOV2	
MAP	P@10	MAP	P@10
$\texttt{LM}_d$ (.204)	f_5^{P-d} (.300)	f_1^{M-d} (.291)	$\texttt{LM}_d$ (.555)
f_7^{M-d} (.196)	f_1^{P-d} (.299)	f_7^{M-d} (.289)	f_7^{P-d} (.544)
$\texttt{LG}_d$ (.194)	$\texttt{LM}_d$ (.293)	$\texttt{LG}_d$ (.288)	f_1^{P-d} (.543)
f_1^{M-d} (.194)	f_4^{P-d} (.292)	$\texttt{LM}_d$ (.280)	f_2^{P-d} (.542)
f_4^{M-d} (.187)	$\texttt{BM}_d$ (.291)	f_2^{M-d} (.274)	f_6^{P-d} (.541)
f_3^{M-d} (.187)	f_6^{P-d} (.287)	$\texttt{BM}_d$ (.274)	$\texttt{LG}_d$ (.541)
f_5^{M-d} (.187)	$\texttt{LG}_d$ (.287)	f_6^{M-d} (.265)	$\texttt{BM}_d$ (.538)
f_6^{M-d} (.186)	f_2^{P-d} (.284)	f_3^{M-d} (.262)	f_5^{P-d} (.535)
f_2^{M-d} (.186)	f_7^{P-d} (.284)	f_4^{M-d} (.261)	f_4^{P-d} (.525)
$\texttt{BM}_d$ (.184)	f_3^{P-d} (.256)	f_5^{M-d} (.260)	f_3^{P-d} (.471)

Table 11. MAP and P@10 measures of different functions and IR models with their default values on WT10G and GOV2 datasets.

WT10G		GOV2	
MAP	P@10	MAP	P@10
$\texttt{LM}_o$ (.473)↓	$\texttt{LM}_o$ (.372)	f_2^{M-o} (.302)	f_2^{P-o} (.557)
f_6^{M-o} (.466)	f_2^{P-o} (.368)	f_1^{M-o} (.295)	f_6^{P-o} (.553)
f_3^{M-o} (.463)	$\texttt{BM}_o$ (.362)	$\texttt{LM}_o$ (.294)	f_1^{P-o} (.551)
f_4^{M-o} (.462)	f_4^{P-o} (.359)	$\texttt{LG}_o$ (.288)	$\texttt{LM}_o$ (.551)
f_5^{M-o} (.454)	f_3^{P-o} (.354)	$\texttt{BM}_o$ (.284)	f_3^{P-o} (.550)
f_7^{M-o} (.453)	f_6^{P-o} (.352)	f_7^{M-o} (.274)	$\texttt{LG}_o$ (.541)
f_2^{M-o} (.449)	f_7^{P-o} (.350)	f_3^{M-o} (.271)	f_7^{P-o} (.539)
f_1^{M-o} (.438)	$\texttt{LG}_o$ (.349)	f_6^{M-o} (.271)	$\texttt{BM}_o$ (.531)
$\texttt{BM}_o$ (.421)	f_1^{P-o} (.347)	f_4^{M-o} (.270)	f_4^{P-o} (.505)
$\texttt{LG}_o$ (.414)	f_5^{P-o} (.337)	f_5^{M-o} (.269)	f_5^{P-o} (.469)

Table 12. MAP and P@10 measures of different functions and IR models with their optimized values on WT10G and GOV2 datasets.

5 Conclusion

In this paper we have addressed the problem of exploring the space of simple IR functions with the goal to discover some promising IR scoring functions. To do so, we proposed a systematic iterative approach to explore the search space till some given length and to identify the set of candidate scoring functions which are mathematically valid and satisfy heuristic IR constraints. We tested the functions obtained on a variety of standard IR test collections.

Our results show that, if one wants to make use of an efficient IR scoring function without tuning parameters on a collection, then one should use:

$$\texttt{ES-LG}(x, y) = e^{\sqrt{\log\left(\frac{x+y}{y}\right)}}$$

where the name ES-LG derives from the fact that this function consists in the exponential of the square root of the log-logistic function. ES-LG is consistently above (for the MAP) all other ones; furthermore, the difference with standard IR functions is significant in several cases. This result is all the more interesting that the "complexity" of this function (measured by its length) is smaller than the one of $\texttt{BM}_d$ and $\texttt{LM}_d$, which can, in theory, be generated by our method using a different term frequency normalization but nevertheless require additional computing resources as they involve more operators. In the situation where it is possible to optimize the value of some parameters (i.e. when relevance judgments are readily available), our results are more contrasted: the difference in ranks between the best functions and the standard ones is not as important as before and if in several cases the difference is significant in favor of the discovered functions, it is, in other cases, in favor of standard functions. All in all, there is no real difference in this case. As ES-LG is ranked second in this setting, we recommend its use in all cases if one is interested in the MAP. It can of course be used as an additional feature in learning to rank approaches.

Acknowledgment. This work was supported in part by the ANR project Class-Y, the Mastodons project Garguantua, and the LabEx PERSYVAL-Lab ANR-11-LABX-0025.

References

1. G. Amati and C. J. Van Rijsbergen. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Trans. Inf. Syst.*, 20(4):357–389, 2002.
2. S. Clinchant and E. Gaussier. Information-based models for ad hoc ir. In *Proceedings of the 33rd ACM SIGIR Conference*, 2010.
3. W. W. Cohen, R. E. Schapire, and Y. Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 10(1):243–270, 1999.
4. K. Crammer and Y. Singer. Pranking with ranking. In *Advances in Neural Information Processing Systems (NIPS 14)*, pages 641–647. MIT Press, 2001.
5. R. Cummins and C. O'Riordan. Evolved term-weighting schemes in information retrieval: an analysis of the solution space. *Artif. Intell. Rev.*, 26(1-2):35–47, 2006.
6. R. Cummins and C. O'Riordan. Evolving local and global weighting schemes in information retrieval. *Inf. Retr.*, 9(3):311–330, 2006.
7. R. Cummins and C. O'Riordan. *Information Extraction from the Internet*, chapter Analysing Ranking Functions in Information Retrieval Using Constraints. CreateSpace Independent Publishing Platform, Aug 2009.
8. R. Cummins and C. O'Riordan. Measuring constraint violations in information retrieval. In *Proceedings of the 32nd SIGIR*, pages 722–723, 2009.
9. W. Fan, M. D. Gordon, and P. Pathak. A generic ranking function discovery framework by genetic programming for information retrieval. *Inf. Process. Manage.*, 40(4):587–602, 2004.
10. H. Fang, T. Tao, and C. Zhai. A formal study of information retrieval heuristics. In *Proceedings of the 27th ACM SIGIR Conference*, 2004.
11. Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 2003.
12. M. Gordon. Probabilistic and genetic algorithms in document retrieval. *Commun. ACM*, 31(10):1208–1218, October 1988.
13. T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the 8th ACM SIGKDD*, pages 133–142, 2002.
14. F. Maes, L. Wehenkel, and D. Ernst. Automatic discovery of ranking formulas for playing with multi-armed bandits. In *Proceedings of EWRL*, pages 5–17, 2011.
15. D. Metzler and W. B. Croft. A markov random field model for term dependencies. In *SIGIR*, pages 472–479, 2005.
16. P. Pathak, M. D. Gordon, and W. Fan. Effective information retrieval using genetic algorithms based matching functions adaptation. In *HICSS*, 2000.
17. J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st ACM SIGIR Conference*, 1998.
18. S. E. Robertson and H. Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.
19. G. Salton and J. McGill. *Introduction to Modern Information Retrieval*. New York,McGraw-Hill, 1983.
20. H. Valizadegan, R. Jin, R. Zhang, and J. Mao. Learning to rank by optimizing ndcg measure. In *Advances in Neural Information Processing Systems (NIPS 22)*, pages 1883–1891, 2009.